**Group: Yingyi**

FIT2099 Assignment 3

Recommendation for change to the game engine

For assignment 2 and 3, we were told to implement our designs without changing anything in the engine code. While implementing our design, we felt that the engine code was well done and we managed to implement our functionalities without changing anything in the engine code. However, there were some problems we discovered in the engine code while doing and we will be discussing the good and bad of the engine code.

**Bad part of the engine code**

**Problem: Item class**

The Item class contains simple implementations where some functionalities were missing which are usually needed in a game. The Item class lacks a higher-level of abstraction where we are forced to downcast the subclass of Item in order to implement the functionality in the subclass. An example would be in the future if an item was given a capability, the player should be able to use that given capability to do something. The item was not given a method to utilize this capability to do something with it where the player has no way to utilize this capability to do something. Therefore, we have to downcast the item to a lower-class level to implement such a method.

**Solving this problem**

This problem can be overcomed by extending the functionality of the Item class. An example would be making the Item class to implement an interface which can be called Usable which contains a method called use(). This would make Item class implement the use() method and can be used by the subclass of Item without the need to use downcasting.

**Advantages**

When incorporating the functionality into the base class, we are utilizing the benefits of polymorphism. This would allow all the subclasses of the base class to inherit or override the method in the base class to implement each of the subclasses' responsibility. This would hugely benefit the flexibility of the code as we are not required to know which subclass we would have to downcast in order to implement the functionality.

**Disadvantages**

By integrating too many functionalities into one base class, the class would have to handle a lot of responsibilities of the subclass where we violate the principle that each class should have their own responsibility. This would mean the base class would become a god class and it is not ideal.

**Good part of the engine code**

**Private Attributes**

By declaring attributes as private, the principle of **“Minimize Dependencies That Cross Encapsulation Boundaries”** are being applied. This means that encapsulation is being applied for information hiding. It is known to be a good practice to declare attributes as private when we create a class. However if we set the attributes to public, this would mean that the attribute is being exposed to the other classes. If in the future, a programmer sets the attribute to private, it is going to break the whole code base and they would have to recode the entire code base. Therefore declaring attributes as private is considered as best practice and the engine code has done that.

**Generic Types**

While inspecting and implementing the engine code, generic types were heavily used. This is seen as a good design practice and implementation as compared to non-generic code as programmers are provided the flexibility to implement generic algorithms. When implementing generic code, programmers can work on many types of collections, easily customizable and provide the ease of code readability. An example would be looking at the codes in Capable class where the methods were implemented with generic types instead of explicitly defining a data type in the parameter of those methods where it would be a tedious job to modify the code if the code smells. Generic class is being used to tell the compiler that the methods are generic which does not consider the actual data type used. To decrease the warnings from a compiler, a <?> wildcard was implemented. Therefore it is seen to be a good practice to use generic types.

**Declaring things to the tightest possible scope**

The engine code has done a good job in reducing the dependencies as much as possible. This can be seen in the GameMap class, in the createMapFromStrings function. We can see that the attributes width and height were not declared at class level but it is declared as a local variable in the method. This would cause a lesser failure in the future where the tighter the scope of something is visible, there would be lesser dependencies on it. The attributes in the class level were also kept together well inside the packages.